I metodi - III

Stefano Mizzaro

Dipartimento di matematica e informatica
Università di Udine
http://www.dimi.uniud.it/mizzaro/
mizzaro@dimi.uniud.it

Programmazione, lezione 11
5 novembre 2007

Comunicazioni di servizio

- Domani non c'è lezione :-(
  - L'ultima con me: lun 12 novembre
- Lezioni di laboratorio:
  - Dal 12 (13) novembre solo 2 fasce orarie:
    - martedì 8:30-10:15
    - mercoledì 14:30-16:15

Dove siamo

- Mattoni, Programmazione strutturata, Sviluppo incrementale, Array
- Intro metodi/sottoprogrammi
  - Definizione (intestazione e corpo) e uso
  - Parametri formali e attuali, associazione
  - Passaggio parametri per valore
  - Funzioni e procedure
  - Utilità dei metodi
  - Durata
  - Visibilità
  - Gestione dei metodi nella JVM
  - Metodi sovraccarichi
  - Funzioni matematiche predefinite

Oggi

- Parametri di tipo array
- Parametri del main
- Esempi
- Ricorsione
  - Definizioni
  - Esempi

Parametri di tipo array

- Sintassi simile ai tipi predefiniti
  - anche per il tipo dei valori restituiti
- Differenze
  - Se si modifica un array in un metodo, non viene modificato solo il parametro formale ma anche il parametro attuale!
  - Passaggio "per riferimento" (in realtà è per valore e viene passato per valore il riferimento)

Esempio

```
class StampaMatrice {

  static void stampaM(int[][] m) {
    for (int i = 0; i < m.length; i++) {
      for (int j = 0; j < m[i].length; j++)
        System.out.print(m[i][j] + "\t");
      System.out.println();
    }
  }

  public static void main (String[] args) {
    int[][] matrice = { {12, 23, 23, 5678},
                        {987, 87 , 3, 0},
                        {12354,34,2,0}};
    stampaM(matrice);
  }
}
```

Esempio

```
class ParametriArray1 {

  static void m(int[] a) {
    for (int i = 0; i < a.length; i++)
      a[i] = 0;
  }

  public static void main (String[] args) {
    int[] vettore = { 1, 2, 3 };
    for (int i = 0; i < vettore.length; i++)
      System.out.println(vettore[i]);
    m(vettore);
    for (int i = 0; i < vettore.length; i++)
      System.out.println(vettore[i]);
  }
}
```

```
1, 2, 3, 0, 0, 0
```

Stefano Mizzaro - Metodi III 7

Diverso da...

```
class Parametri {

  static void inc(int x) {
    x++;
  }

  public static void main (String[] args) {
    int y = 0;
    inc(y);
    System.out.println(y);
  }
}
```

```
>javac Parametri.java
>java Parametri
0
>
```

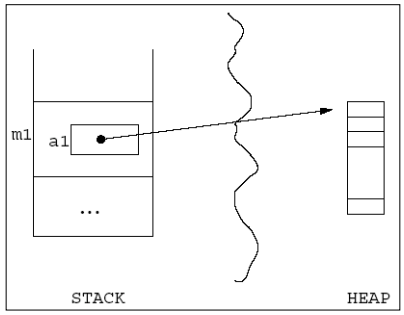
Stefano Mizzaro - Metodi III 8

Perché

- Passaggio per (valore del) riferimento
- Le variabili e i parametri di tipo primitivo stanno sulla pila dei record di attivazione
- Gli array no, stanno sullo heap
  - Zona di memoria separata dalla pila e gestita in modo più "disordinato"
- Sullo stack ci stanno
  - I valori delle variabili di tipo primitivo
  - I riferimenti agli array
    - Riferimento ~ indirizzo in memoria

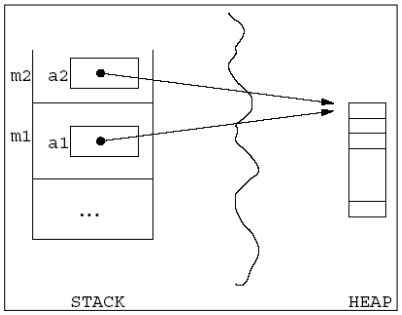
Stefano Mizzaro - Metodi III 9

Perché



Stefano Mizzaro - Metodi III 10

Perché



Stefano Mizzaro - Metodi III 11

Esempio rivisto

```
class ParametriArray1 {
  static void m(int[] a, int b) {
    for (int i = 0; i < a.length; i++)
      a[i] = 0;
    b = 0;
  }
  public static void main (String[] args) {
    int[] vettore = { 1, 2, 3 };
    int x = 4;
    for (int i = 0; i < vettore.length; i++)
      System.out.println(vettore[i]);
    System.out.println(x);
    m(vettore, x);
    for (int i = 0; i < vettore.length; i++)
      System.out.println(vettore[i]);
    System.out.println(x);
  }
}
```

```
1, 2, 3, 4, 0, 0, 0, 4
```

Stefano Mizzaro - Metodi III 12

E gli array di array?

- Un array è un array
- Un tipo primitivo è un tipo primitivo
- Ergo, se l'elemento di un array è:
  - di un tipo primitivo, viene passato per valore
  - di tipo array, viene passato per (valore il) riferimento
- Esercizio
  - Scrivere un metodo void **`stampaM(int[][] m)`** che:
    - visualizza la matrice `m`;
    - invoca un metodo **`void azzera(int [] a)`** per azzerare il vettore in `m[0]`

Quindi, riassumendo

- I parametri di tipo predefinito (**`byte`**, **`short`**, **`int`**, **`long`**, **`float`**, **`double`**, **`char`**, **`boolean`**) vengono passati per valore
  - Le modifiche non si ripercuotono sul parametro attuale
- I parametri di tipo array sono passati "per riferimento" (o meglio, il loro riferimento è passato per valore)
  - Modifiche su parametro formale → modifiche su parametro attuale
  - (anche gli oggetti, di cui parleremo…)

Esempio

- Riprendiamo l'esempio dei numeri complessi
  - Un numero complesso rappresentato con un array di due posizioni
  - (meglio di 2 variabili, gestione più unitaria)
  - Sfruttiamo
    - Passaggio parametri di tipo array
    - Passaggio per valore del riferimento
  - Un po' meglio, ma…
  - … si potrà fare ancora molto meglio!

I numeri complessi (1/2)

```
class Complessi {
  static double[] prodC (double[] a, double[] b){
    double[] c = new double[2];
    c[0] = a[0] * b[0] - a[1] * b[1];
    c[1] = a[0] * b[1] + a[1] * b[0];
    return c;
  }
  static double[] prodC (double[] a, double[] b,
                         double[] c) {
    return (prodC(prodC(a,b),c));
  }
  static void leggiC(double[] a) {
    System.out.print("Parte reale: ");
    a[0] = Leggi.unDouble();
    System.out.print("Parte immaginaria: ");
    a[1] = Leggi.unDouble();
  }
  static void scriviC(double[] a) {
    System.out.print("("+a[0]+"+i*"+a[1]+")");
  }
```


I numeri complessi (2/2)

```
public static void main (String[] args) {
    double[] x = new double[2];
    double[] y = new double[2];
    double[] z = new double[2];
    double[] w = new double[2];
    leggiC(x);
    leggiC(y);
    leggiC(z);
    w = prodC(x,y,z);
    scriviC(x);
    System.out.print(" * ");
    scriviC(y);
    System.out.print(" * ");
    scriviC(z);
    System.out.print(" = ");
    scriviC(w);
    System.out.println();
  }
}
```

`scriviC(prodC(x,y,z));`

Parametri del main

- Il main è un metodo
  - parametri formali: un array di **`String`**
  - parametri attuali specificati all'invocazione della JVM

```
class Main {
  public static void main (String[] args) {
    System.out.println("Numero argomenti: " +
                       args.length);
    for (int i = 0; i < args.length; i++)
      System.out.println(args[i]);
  }
}
```

```
>java Main
Numero argomenti: 0
>java Main pippo pluto minni
Numero argomenti: 3
pippo
pluto
minni
>
```

Scaletta

- Parametri di tipo array
- Parametri del main
- Esempi
- Ricorsione
  - Definizioni
  - Esempi

Ricorsione

- Definizione ricorsiva in matematica:
  - Definizione di un concetto usando il concetto stesso
- Definizione ricorsiva in Java:
  - Definizione di un metodo usando il metodo stesso

Ricorsione in matematica

- Somma (+)
  - x + 0 = x
  - x + succ(y) = succ(x + y)
    (succ è il successore)
- Fattoriale
  - 0! = 1
  - n! = n * (n – 1)!
- Sono definizioni operative
  - 4 + 3 = 4 + succ(2) = succ(4 + 2) …
  - 5! = 5 * 4! = 5 * …
- Caso base e passo

Def. ricorsiva vs. circolare

- Somma (+) ricorsiva:
  - x + 0 = x
  - x + succ(y) = succ(x + y)
- Somma (+) circolare:
  - x + 0 = x
  - x + y = pred(succ(x) + y)
  - Def. "corretta", ma non operativa:
    - 4 + 3 =
      pred(succ(4)+3) =
      pred(pred(succ(succ(4))+3)) =
      pred(pred(pred(succ(succ(succ(4)))+3))) = …

Funzione di Fibonacci

- F(0) = 1
- F(1) = 1
- F(n) = F(n – 1) + F(n – 2)

- F(5) = F(4) + F(3) = …

Algoritmi ricorsivi

- Anche gli algoritmi possono essere definiti ricorsivamente
- Es.: date N (=3^m) palline, di cui una più pesante, e una bilancia a 2 bracci, trovare la pallina pesante
  - Divido in 3 gruppi equinumerosi
  - Ne peso 2
  - Se ==, li scarto e scelgo il 3o gruppo, altrimenti scelgo il gruppo più pesante. Poi ripeto ricorsivamente sul gruppo selezionato (che avrà 3^{m-1} palline…)

Fattoriale in Java

- Metodo ricorsivo per il calcolo del fattoriale
  - 0! = 1
  - n! = n * (n – 1)!

```
static int fatt(int n) {
  if (n == 0)
    return 1;
  else
    return n * fatt(n - 1);
}
```


Fibonacci in Java

- Metodo ricorsivo per il calcolo della funzione di Fibonacci
  - F(0) = 1
  - F(1) = 1
  - F(n) = F(n – 1) + F(n – 2)

```
static int F(int n) {
  if (n == 0 || n == 1)
    return 1;
  else
    return (F(n - 1) + F(n - 2));
}
```


Perché la ricorsione funziona

- La ricorsione e la pila dei record di attivazione
  - Ogni invocazione/esecuzione di metodo ha un'allocazione di un record di attivazione sulla pila della JVM
  - I record di attivazione allocati mantengono i risultati intermedi. Vediamo un es.

```
class Fattoriale {
  static int fatt(int n) {
    int res = 0;
    if (n == 0)
      res = 1;
    else
      res = n * fatt(n - 1);
    return res;
  }
  public static void main (String[] args) {
    int x = 4;
    System.out.println(fatt(x));
  }
}
```


Ricorsione mutua (indiretta)

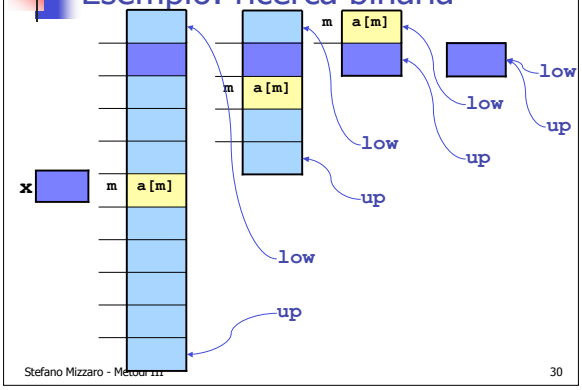
- Un metodo **`m1`** ne chiama un altro **`m2`** (che ne chiama un altro **`m3`** …) che chiama **`m1`**
- Es. (giochino): Definiamo, usando la ricorsione mutua, due metodi **`pari`** e **`dispari`**
  - Idea:
    - pari(n) = dispari(n-1)
    - dispari(n) = pari(n-1)
  - Base della ricorsione:
    - pari(0) = true
    - dispari(0) = false

Il codice

```
static boolean pari(int n) {
  if (n == 0)
    return true;
  else
    return dispari(n - 1);
}

static boolean dispari(int n) {
  if (n == 0)
    return false;
  else
    return pari(n - 1);
}
```


Esempio: ricerca binaria



Lo pseudocodice

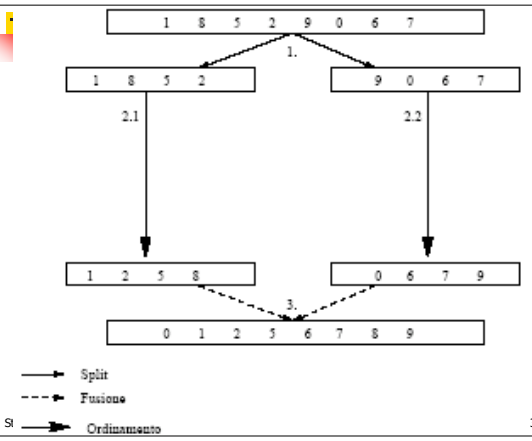
- Cerco in tutto l'array:
  - Individuo l'elemento mediano m
  - Se x == m, ho finito
  - Se x > m cerco (ricorsivamente) nella seconda metà
  - Se x < m cerco (ricorsivamente) nella prima metà

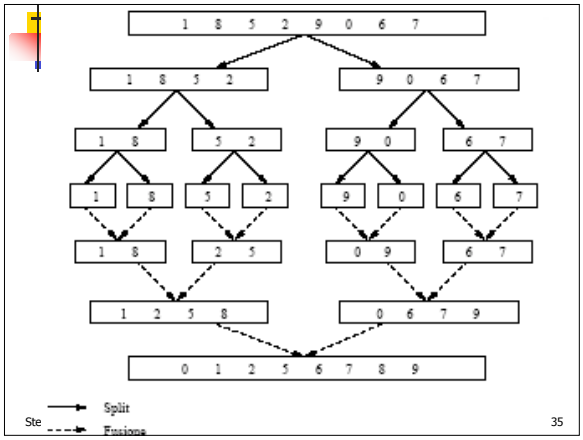
Il codice

```
static int ricerca (char[] a, char x,
                    int low, int up) {
  int m;
  m = (up + low) / 2;
  if (up < low)
    return -1;
  else if (a[m] == x)
    return m;
  else if (a[m] < x)
    return ricerca(a, x, m + 1, up);
  else //a[m] > x
    return ricerca(a, x, low, m - 1);
}
```


Esempio: ordinamento per fusione (*merge sort*)

- Ragioniamo in modo ricorsivo
  - 1. **spezza** l'array a metà;
  - 2.1. **ordina** la prima metà;
  - 2.2. **ordina** la seconda metà;
  - 3. **fonde** le due metà
- La ricorsione è sull'operazione **ordina**
  - che viene effettuata su un pezzo di array più corto…
  - …che verrà ordinato con lo stesso procedimento!
- C'è anche la fusione…





Il codice (1/2)

- Non servono array ausiliari
- Ipotizziamo lunghezza = 2^n

```
/** Ordina l'array a da l a u */
  static void ordina(char[] a, int l, int u) {
    int m;
    if (l != u) {
      m = (l + u) / 2;
      ordina(a, l, m);
      ordina(a, m + 1, u);
      fondi(a, l, m, u);
    }
  }
```

Il codice (2/2)

```
/** Fonde l'array a, da l a m e da m + 1 a u
*/
static void fondi (char[] a,int l,int m,int u){
  char[] b = new char[u - l + 1];
  int i = l, j = m + 1, k = 0;
  while (i <= m && j <= u)
    if (a[i] <= a[j])
      b[k++] = a[i++];
    else
      b[k++] = a[j++];
  while (i <= m)
    b[k++] = a[i++];
  while (j <= u)
    b[k++] = a[j++];
  for (k = 0; k <= u - l; k++)
    a[k + l] = b[k];
}
```


Sulla potenza della ricorsione

- Espressività
  - A volte la definizione più semplice e naturale di un algoritmo è ricorsiva
  - Es.: algoritmi di visita su alberi… ASD…
  - Le istruzioni di iterazione e la ricorsione sono equivalenti!
- È difficile
  - domanda x lode
  - ne riparlerete

Esercizi

- Metodo ricorsivo per calcolare il MCD con l'algoritmo di Euclide
- Metodo ricorsivo per assegnare `i` ad `a[i]`
  - (suggerimento: usare un parametro ausiliario)
- Metodo ricorsivo per invertire un array
- …

Riassunto

- Metodi
  - Parametri di tipo array
  - Parametri del main
  - Esempi
  - Ricorsione
- Libro: fino a cap. 6
- Eserciziario: fino a § 3.2
- Prossima lezione (ultima con me)
  - Computabilità
  - Esercizi